Plug-and-Play Interactive Deep Network Visualization

Gjorgji Strezoski*
Informatics Institute, University of Amsterdam

Marcel Worring†
Informatics Institute, University of Amsterdam

ABSTRACT

Deep models are at the heart of computer vision research recently. With a significant performance boost over conventional approaches, it is relatively easy to treat them like black boxes and enjoy the benefits they offer. However, if we are to improve and develop them further, understanding their reasoning process is key. Motivated by making the understanding process effortless both for the scientists who develop these models and the professionals using them, in this paper, we present an interactive plug&play web based deep learning visualization system. Our system allows users to upload their trained models and visualize the maximum activations of specific units or create attention/saliency maps over their input. It operates on top of most popular deep learning frameworks and is platform independent due to its web based implementation. We demonstrate the practical aspects of our two main features MaxOut and Reason through visualizations on models trained with artistic paintings from the OmniArt dataset and elaborate on the results.

Index Terms: H.2.8 [[Data Visualization] Convolutional Neural Networks;]: Visual Analytics—Deep Learning;

1 INTRODUCTION

Deep neural networks have been the base mechanism for solving computer vision problems in recent years [8, 12, 13, 16, 17]. These models are constructed of multiple layers, containing hundreds, even thousands of processing units with the ability to learn millions of parameters. This layered structure and the ability to work in large high dimensional spaces is what gives deep models superior performance in solving complex computer vision problems. However, their greatest strength is also one of their greatest weaknesses - the reasoning process of a deep model is not easily interpretable. Various parameters, feature maps and weight matrices give hints about the decision making process, but often the reasons behind a decision made by the model remain a mystery. A clear understanding of the learned features and representations is important for both the people who develop these models and the professionals who use them.

On one hand, by using interactive visualizations, scientists can better observe the strengths and weaknesses of their model, on the other, end users can acquire new insights from the purely objective viewpoint of a deep model. To achieve this we address two fundamental questions posed in deep learning:

1. How does the model perceive a certain target? What would the *perfect* image for the target look like?
2. Given an input image, which regions are responsible for the decision the model has made when assigning the target category?

Answering these questions can reveal a lot about how the model in question functions and how decisions are made. For this reason, we attempt to answer those questions through a web based tool that contains two main features, namely:

1. **MaxOut**: Given a trained model, we maximize the activations for a particular class (in classification problems) or a particular value (in regression problems). Activations can be computed for every unit throughout all the layers in the model given a desired or predicted target.
2. **Reason**: Given a trained model we apply class activation mapping techniques to map target specific regions in the input image. This procedure is essentially answering the question, why the model classified the input image in a specific category and which part of the image is most responsible for this decision.

Both scenarios can be used at the same time, on the same model, running on both CPU and GPU, supporting multiple deep learning frameworks and back-ends.

Applying deep model visualization techniques is often a complex process and requires both knowledge in the machine learning and software engineering domain. Motivated by making deep model visualizations accessible to everyone who uses a deep model in their domain, we created this generic tool for visualizing the internals of a deep net. This symbiosis between visual analytics, computer vision and user interaction is particularly interesting in the art domain as it allows art historians, curators, professionals and even students to analyze artistic data from a completely new perspective. We use the art domain to demonstrate our two features and show an interesting use case for our tool.

In the list below, we indicate the main contributions of this paper:

- We developed a plug&play, interactive visualization system for deep models running in a web browser with no dependency on the training platform.
- We incorporated state-of-the-art visualization techniques for deep models available in a user friendly environment, where no coding is necessary to produce the desired result.
- We developed an extension to the Grad-CAM method that allows for attention maps to be computed with respect to multiple layers at the same time.

2 RELATED WORK

An ideal system does not exist. There is always a trade-off between different features in the same system and in order to excel in one, compromises must be made. The current state of machine learning research portraits the same picture. Classical rule based system are highly interpretable, however they lack robustness and accuracy [14]. On the other side, deep models obtain remarkable performance due to the increased number of learnable parameters and closed loop end-to-end training processes, however their decision making process is very hard to interpret. The high dimensional space in which deep models operate can be perceived as an information overload and the goal of visual analytics research is to turn this information overload into an opportunity for gaining new insight [10].

Visualizations in this domain are generally driven by either the visual information incorporated in the structure of the underlying model, or by the correlation between the model's output and the input data, or some of the learned internal representations. Nevertheless, the common goal in all of them is to make deep models more interpretable and their reasoning process understandable.

*e-mail: g.strezoski@uva.nl
†e-mail: m.worring@uva.nl

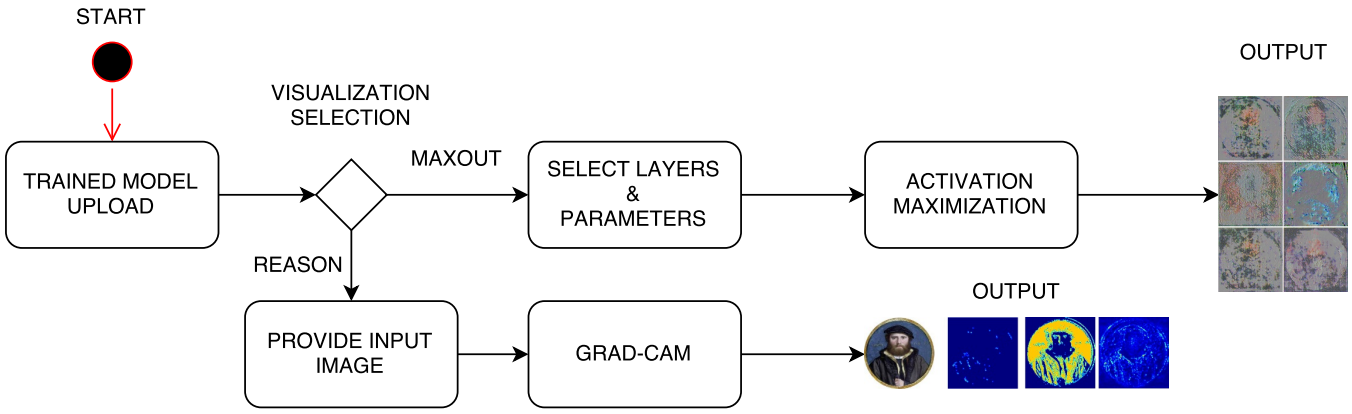

Figure 1: User interaction flow in our tool in both the Reason and MaxOut use-case scenarios.

2.1 Understanding Deep Models

Many research efforts have analyzed the internal representation of data and units in deep networks [6, 15, 19, 24–26].

Liu et al. [15] proposed a novel visualization model, where they represent a convolutional neural network as a directed acyclic graph in which they visualize the interaction between multiple facets of each neuron and their interactions. Further, using a rectangle packing algorithm and matrix reordering they are also able to show the different derived features in the intermediate layers during training. While the approach of Liu et al. projects the visualization in an external space, Zhou et al. [25] introduce an approach called Class Activation Mapping (CAM) that projects its insight over the input space. By altering the architecture of deep convolutional models with adding a global average pooling layer, they are able to preserve the spatial information that is lost in its fully connected top layers. Adding the global average pooling layer further increases the localization abilities of the model and outputs spatial coverage maps over the input image for a selected label. However, this approach does imply changing the model's structure and adding an additional layer which will require re-training. Selvaraju et al. [19] overcame the problem of changing the model's structure with Grad-CAM, an approach to Class Activation Mapping using the computed gradients as the back-propagated signal. By setting the gradients for all categories (excluding the desired/predicted one) to zero and back-propagating that information to the desired layer, a coarse localization map is computed. Using guided pack-propagation [21] the coarse localization map is then point-wise multiplied with the guided back-propagation output to obtain a concept specific visualization.

In our approach, we incorporate the Grad-CAM methodology as it does not require architectural changes to the model in order to produce the visualizations and further extend it to be applicable to a set of layers, instead of just one. This is particularly useful for the low level features learned in the early layers of a convolutional neural network, which have a rather uniform activation map throughout the input image.

All of the approaches above focus on visualizing insights from an already trained model. In contrast, Smilkov et al. [20] show a high level abstraction visualization over the learning process in an interactive visualization of a user customizable deep neural network. The process of training models in the Smilkov et al. system is limited in terms of data and complexity due to the web browser's resource stack, but nevertheless it provides real-time information about the learning process. In the paradigm of visualizing the training process of a deep model, control through visualization is also possible. As Chung et al. [4] demonstrate in ReVACNN through visualizing the two dimensional embedding space of convolutional filters and weights with user controlled steering of the model. They achieve control by adding or removing nodes and setting custom weights. Another example of understanding deep models through visualization is a tool box that has been developed by Yosinski et al. [23] running on the Caffe back-end. This toolbox displays a neuron by neuron visualization of each layer in the convolutional network in a video feed real time.

In this paper we incorporate most of the above approaches into one interactive plug&play web based system that runs on all platforms with support for multiple deep learning frameworks in both CPU and GPU operation modes. This system can be used as a web service or local instance running on any machine.

3 System Architecture

We introduce a generic web based visualization tool for visualizing the reasoning process in a deep model. The system's implementation logic spans on both the user (front-end) side and server (back-end) side. In the front-end the system parses the uploaded model and configures the back-end with respect to the model's operating parameters like source framework, CPU/GPU configuration, layer settings and also available system resources. On the server side, a model specific sub-server instance is created that feeds data through a web hook to an Model-View-View-Model (MVVM) controller on the front-end which then displays the data.

With deep models there is always the possibility to store them as a dictionary of layers and weights. This structure is something that all the deep learning frameworks have in common, so moving from one framework to another is just a matter of different structure parsing. For our implementation we convert each uploaded model to Keras with Tensorflow on the fly and store the new structure in the local file-system. In this way we preserve the code-base integrity and we are able to use the same visualization functions regardless of the deep learning framework used to train the model.

The general user interaction flow is illustrated in Figure 1 for both the MaxOut and Reason feature.

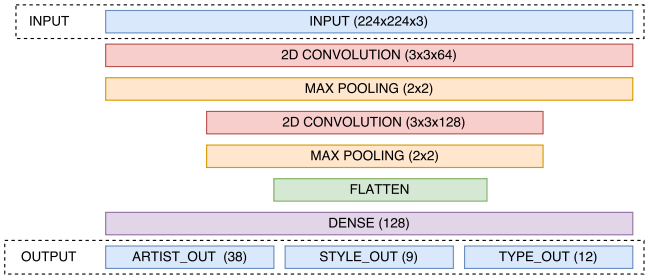

Figure 2: A high level model overview for the artistic data use-case.

Table 1: Dashboard parameter functionality description for MaxOut and Reason

Parameter	Type	Description
Filter #	Integer [0 - # Units in layer]	Select the filter unit/s to visualize. If not set all units are selected. (MaxOut & Reason)
Class #	Integer [0 - # Targets]	Select a specific class to maximize (MaxOut) or calculate attention maps (Reason)
TV Weight	Float [0 - 100]	Weight parameter for the Total Variation loss (MaxOut)
Norm	Float [0 - 100]	Weight parameter for the regularization loss (MaxOut & Reason)
Iterations	Integer [Unbound]	Number of iterations for maximum activation input image generation (MaxOut)
Smoothing	Boolean	Apply smoothing to generated maximum activation images (MaxOut) / Smooth out heat-maps over the high resolution view (Reason)

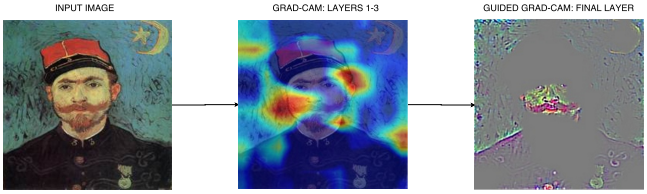

Figure 3: Reason: Grad-CAM over a collection of layers and Guided Grad-CAM in the last fully connected layer showing different relevant regions for the Van Gogh class in a model trained for artist attribution. An interesting observation is that higher level layers focus on complete objects like the mustache, nose and mouth regions while lower level layers find the contours of the object and some background artifacts more interesting.

Table 2: Task specific data information

Task	Type	# Classes	# Samples
Artist Attribution	Classification	38	
Style Prediction	Classification	9	43000
Type Prediction	Multi-label Classification	12	

3.1 Back-End

With the expansion of the deep learning paradigm, many deep learning frameworks have been developed like Caffe [9], Tensorflow [1], CNTK [18], Theano [2]. In our system we currently support models built on top of Tensorflow and Theano, even if they were made with a high level abstraction library like Keras [3] or Lasagne [5].

In terms of the web server, we use the Django web framework with a MongoDb engine for storing results and visualizations. Our back-end implementation additionally features a Redis smart server caching engine for speeding up high dimensional model visualizations when repetitive requests are made. Because some of the visualizations that can be generated in our tool require training iterations, we support GPU operation modes on a local instance.

3.2 Front-End

On the client side we incorporate an MVVM software architecture with AngularJS and jQuerry for handling data operations and moderating the server feed, while for displaying specific visualizations we use D3 and HTML5 Canvas. The major role of the front-end is to prepare the configuration for creating the back-end instance responsible for the current session.

4 Methods

Our tool incorporates two main features, namely MaxOut and Reason. MaxOut is a tool for displaying the maximum activation input for a specific target in any layer of the uploaded model. Reason on the other hand, is a tool for creating class specific attention maps, also in any layer of the uploaded model. Attention maps can be created for a single layer or multiple layers at the same time.

Both features share a common dashboard where users can control the parameters for creating the visualizations. The functionality and possible values per parameter are explained in Table 1.

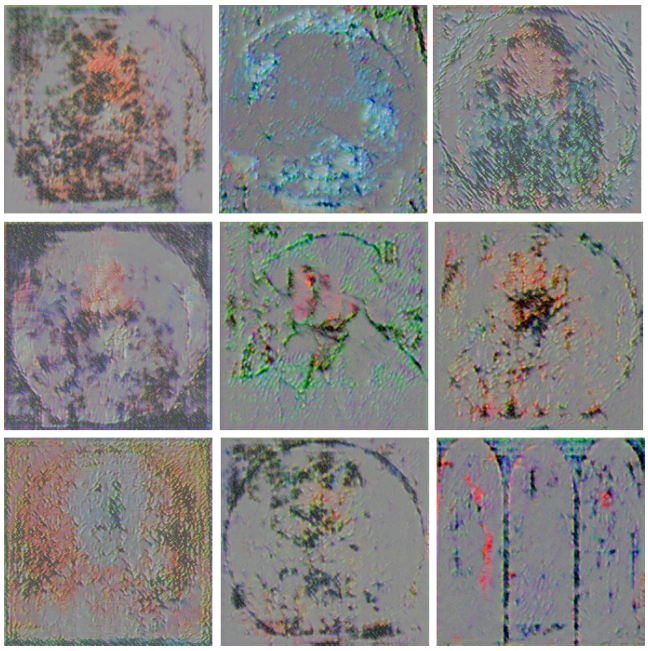

Figure 4: Maximum activations in a penultimate dense layer of a convolutional neural network for a portrait type in the grid view. Clearly visible portrait outlines are present in almost all of the visualized units. Some of the outputs even have distinguishable facial features.

4.1 MaxOut: Visualizing Maximum Activations

In a CNN, each convolutional layer has several learned template matching filters that maximize their output when a similar template pattern is found in the input image. Usually the first convolutional layer is easy to interpret as it contains primitive features like edges, lines and color transitions. To see what the convolutional layer is doing, a simple option is to apply the filter over raw input pixels and multiply to get a result. Subsequent convolutional filters operate over the outputs of previous convolutional filters (which indicate the presence or absence of some templates), making them much harder to interpret.

Having in mind that a simple weight visualization does not provide enough information for understanding the structure of the intermediate layers, activation maximization is an intuitive approach to visualize how they react. The idea behind activation maximization is to generate an input image that yields the maximum activation scores from the units in a particular layer. This is performed using a custom loss function for a user defined number of iterations which outputs small values for significant filter activations. Using this feature helps interpret how the model in question perceives the classes on which it is trained on, in different levels. When this feature is used in the last layers in the model's structure, observations show that a mapping between target categories and specific units can be deduced. Figure 5 illustrates the flow of actions performed in the MaxOut feature from start to end.

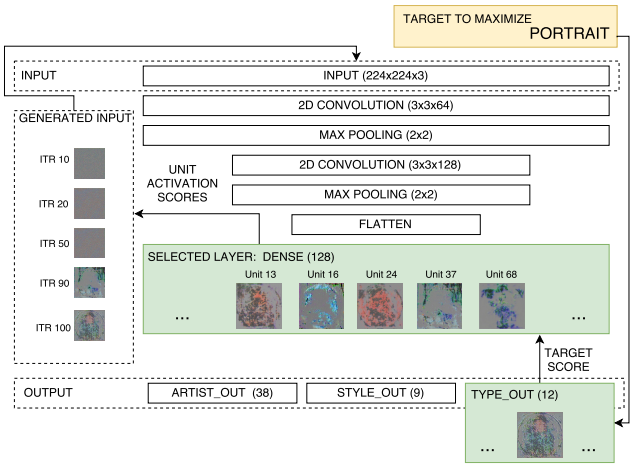

Figure 5: An illustration of the MaxOut process. The desired target and layer whose units are to be maximized are selected in the beginning. With respect to this selection, the generated input images (left dotted square) progress iteratively as the loss for the category per unit decreases.

4.2 Reason: Visualizing Attention Regions

Reason is a feature where we visualize which regions of the input image are relevant to a particular class, according to layers belonging to different levels in the model. We perform this using GradCAM with and without guided back-propagation. Given an input image, we use the uploaded model to predict the target and visualize the regions that most contributed to the result in every parametrized layer. We further enhance user interaction by providing the user with an option to select groups of layers to visualize, since early convolutional layers contain low level image features that have a more uniform activation distribution than layers appearing later in the architecture.

Figure 3 shows the difference between a regular GradCAM output and a GradCAM output with Guided Back-propagation. The regular GradCAM visualization generates heat-maps over the input image and localizes well, however these heat-maps lack the ability to show fine-grained importance which is possible with pixel-space gradient visualization methods (Guided Back-propagation and Deconvolution). For this reason we only apply the regular GradCAM to visualizations in layers lower in the architecture, while for ultimate and penultimate layers we apply GradCAM with Guided Back-propagation. Figure 6 illustrates the flow of data and representations generated by the Reason feature.

The generated regions in each image do not have to relate to the target category only. This feature is particularly useful in situations when the model does not yield high confidence scores for the final output. For example, in the artistic domain it can be useful to determine confusing regions in artist and style attribution, as well as analyzing feature dependencies between styles.

5 Use Cases

Visualizing the internal representations and decision making process of a deep model can generate new insight and answer some domain specific questions. We are particularly interested in the artistic domain, so we applied our tool to models trained with a dataset of artistic paintings from the early 1500s to late 1900s with the artists, painting type and style as targets. Given the multiple types of tasks (style, artist, type) we performed multi-task learning with a shared fully connected layer before the final classification block for each task. With intent of creating domain specific visualizations of the internal states and attention maps, we avoided pre-trained models even if they have superior performance to models trained from scratch. We trained our models from scratch on artistic data consisting of

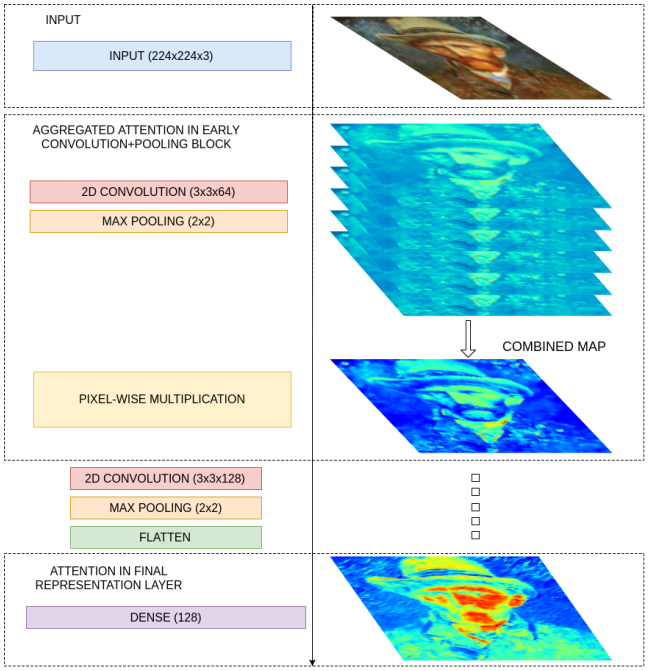

Figure 6: Attention representation flow in the Reason feature. The attention maps from the early layers in the convolutional architecture are combined to form a single aggregated attention map that conveys a clearer image of important regions. In the final representation layer before the classification block, a much stronger heat-map is generated due to the higher level target specific features.

43,000 paintings from OmniArt [22] for artist attribution, style and type prediction. Task specific dataset details are given in Table 2.

For the model illustrated in Figure 2, the training process lasted for 40 epochs with a batch size of 64. For optimization we used the Adam optimizer [11] with an initial learning rate of 0.001. Training on a single TitanX GPU, the total training lasted 5 hours. The final task specific performance of the model is presented in Table 3

5.1 MaxOut in Artistic Data

A model can learn to distinguish between different artists, artwork types, styles and periods. It can even learn the particular characteristics of a certain style, store them in a gram matrix and then apply in a different input space [7]. This means that the models that we train build their own representation of what it means for an artwork to be a portrait, or a painting to belong to the Realism style. While this is a well known mechanism for people with a background in computer vision or machine learning, professionals in the artistic domain do not have the skill-set required to use these methods and study the occurring phenomenons.

Using the MaxOut feature, seeing the *ideal* depiction of a certain target is possible with just a few clicks. As a simple, yet illustrative example we applied the MaxOut feature to a model trained to distinguish painting types (portraits, landscapes, still nature, etc.). When choosing the desired target painting type to be a portrait, the MaxOut feature outputs images that have a rather familiar composition. This is illustrated in Figure 4 for portraits and Figure 7 for landscapes. For the units correlated with the portrait target, all of the generated inputs have a central structure resembling a human head with facial features, hair and even hats. Another rather interesting find is that most of the portraits in out dataset are painted on a circular shaped canvas, which is also captured in the internal representation of the model for a portrait. Going into further detail using the pan&zoom feature, interesting color patterns can be observed for illustrating skin tones.

Table 3: Task specific model performance

Task	Type	Score	Metric
Artist Attribution	Classification	0.74	Accuracy
Style Prediction	Classification	0.88	
Type Prediction	Multi-label Classification	0.91	image MAP

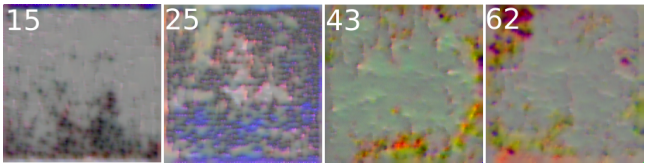

Figure 7: Maximum activation in the penultimate dense layer correlated with the *Landscape* target. Blue pigments on the bottom indicate presence of water and blue top regions correlate to skies, making units 15 and 25 prone to activate on coastal landscapes. Green, yellow and orange colors in units 43 and 62 indicate verdant landscapes.

5.2 Reason in Artistic Data

When viewing a particular painting, one can determine its origins, style, period and even artist from clues hidden in the composition and structure of the artwork. People's perspective on this is often well defined, for example heavy dramatic brush strokes might indicate a Van Gogh, or careful light capturing and low lit dark tones might indicate a Monet. Due to the nature of human perception, representation of art is always open to interpretation. We can also pose this question in the world of neural networks. What the neural network thinks about when classifying a Van Gogh painting is a particularly interesting process to visualize. Also, where does the network pay the most attention when deciding whether that particular artist should be attributed?

We can answer these questions by analyzing target specific regions in artworks. This sort of analysis requires a high resolution visualization, however the receptive field of convolutional neural network is relatively small for this purpose (max. 256px over 3 channels). For this reason after applying the guided back-propagation on the scaled input image, we further interpolate the same regions over the input image in its original dimensions providing us with real size attention maps. Additionally, while displaying a full screen grid preview of the maps from each layer there is a synchronized pan and view functionality over all maps, allowing parallel exploration of the visualization results. This detailed view can be accessed from the Reason and MaxOut dashboards after creating the attention maps or maximum activations. From Figure 3, we can conclude that the network finds Van Gogh specific artifacts in the facial features like mouth, moustache and nose area in the final layers, while the earlier layers in its structure indicate some contour and background features as target relevant. This also confirms that the higher level layers, learn more semantically relevant features than lower level layers.

Additionally, Figure 6 shows a similar insight, where the earlier layers in the architecture activate more uniformly on the input, so the activations are less likely to be class specific. In the final part of Figure 6 we can again see that what makes the input image a Van Gogh self portrait are the actual facial features captured in the final representation before the classification block.

Figure 8 offers another interesting insight into our model. In the first row we show a maximum activation of a unit correlated with some landscape images from our dataset. The dark lower portion matches well with Rembrandt's dark tones and gray skies in stormy landscapes. When one of these landscape images in run through the Reason feature, we can clearly see that according to our model, the sky portion of the image is the one responsible for attributing both Rembrandt as the artist and landscape as the type. This is visible for the two input images in the second and third row in Figure 8, where the red box represents Reason in artwork

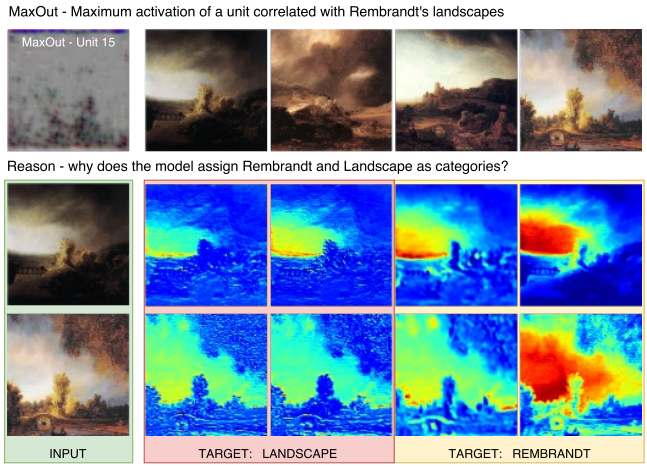

Figure 8: MaxOut and Reason in Rembrandt's landscapes. In the first row we show a maximum activation of a unit from the final dense representation that has a minimum distance to Rembrandt's landscapes. Second and third row show why these particular paintings are attributed the Landscape type (in red) and Rembrandt artist (in yellow).

type prediction and the yellow box represents Reason in the artist attribution process. Inspecting the dataset after the fact, reveals that most images correlated with both the Rembrandt and Landscape target have the same type of sky in the mid-top left portion of the painting. This would imply that the model learned a suitable representation for this subset of images and truly captured target specific knowledge.

6 Conclusion

In this paper we introduced a cross-platform, plug&play deep visualization tool which we applied on models trained on artistic paintings. This system is plug&play, meaning that no configuration is necessary for running it and being platform independent can be used to visualize models trained with a variety of frameworks with different settings. We put our main emphasis on ease of use and simplicity because the tool is intended to be a fast visualization solution requiring a minimum amount of effort to configure.

MaxOut and Reason, the two main features focus on different perspectives on visualizing the model's reasoning. MaxOut creates maximum activation input images for a specified target enabling users to see the internal representation of that target in any layer in the network. This feature could answer interesting questions in the artistic domain as it emphasizes the layer relevant features in the generated maximum activation images. Depending on the level of the selected layer, analysis can be performed on either low level basic features containing brush strokes, contours and simple color gradients, or high level features containing target specific objects like facial features.

Reason generates attention maps over an input image with respect to a certain layer, or even a layer collection, for a specified or predicted target. The specified targets and layers can be aggregated for generating a combined attention view and comparing attention regions on the input image according to multiple layers. Both features have a grid based, high resolution view of the generated output with synchronized pan and zoom features.

This tool can be used as a web service available on-line, or as a local instance on a private computer. Making it available as web service, requires a powerful server for serving multiple clients at once, so currently we can only release the source code and use it in a local environment. We will further develop and improve this tool by supporting more types of visualizations, improving the support for Caffe and CNTK back-ends and the general user experience.

ACKNOWLEDGMENTS

This research is part of the VISTORY project supported by NWO (Netherlands Organization for Scientific Research) through NICAS (Netherlands Institute for Conservation, Art and Science).

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pp. 1–7, 2010.

[3] F. Chollet et al. Keras, 2015.

[4] S. Chung, S. Suh, C. Park, K. Kang, J. Choo, and B. C. Kwon. Revacnn: Real-time visual analytics for convolutional neural network.

[5] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, et al. Lasagne: first release. *Zenodo: Geneva, Switzerland*, 2015.

[6] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, June 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678. ACM, 2014.

[10] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pp. 9–16. IEEE, 2006.

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[12] I. Kokkinos. Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *arXiv preprint arXiv:1609.02132*, 2016.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[15] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[18] F. Seide and A. Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2135–2135. ACM, 2016.

[19] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *arXiv preprint arXiv:1610.02391*, 2016.

[20] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, and M. Wattenberg. Direct-manipulation visualization of deep networks. In *Workshop on Visualization for Deep Learning of the 33rd International Conference on Machine Learning*, 2016.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[22] G. Strezoski and M. Worring. Omniart: Multi-task deep learning for artistic data analysis. *arXiv preprint arXiv:1708.00684*, 2017.

[23] J. Yosinski, J. Clune, A. M. Nguyen, T. J. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015.

[24] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

[25] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[26] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.